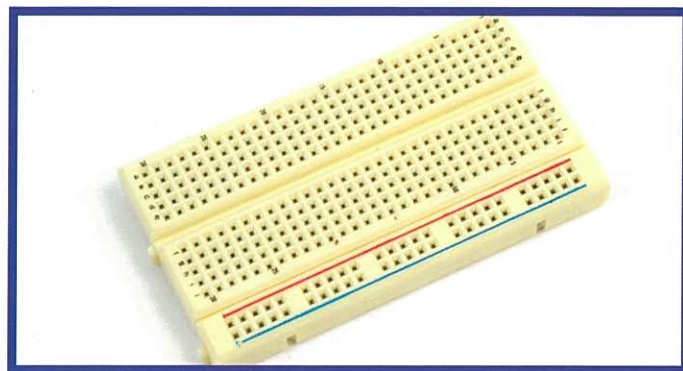
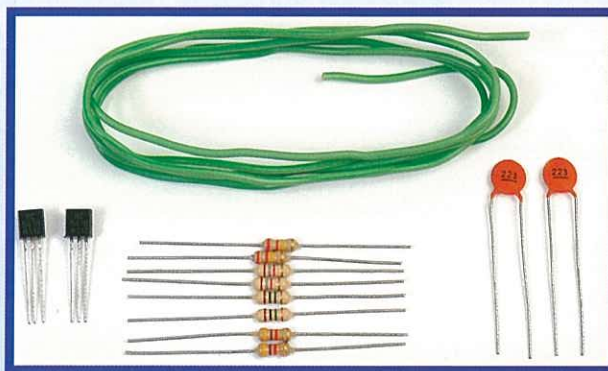
Direttore responsabile:
ALBERTO PERUZZO
Direttore Grandi Opere:
GIORGIO VERCELLINI
Consulenza tecnica e traduzioni:
CONSULCOMP S.n.c.
Pianificazione tecnica
LEONARDO PITTON

Direzione, Redazione, Amministrazione: viale Ercole Marelli 165, Tel. 02/242021, 20099 Sesto San Giovanni (Mi). Pubblicazione settimanale. Registrazione del Tribunale di Monza n. 1738 del 26/05/2004. Spedizione in abbonamento postale gr. II/70; autorizzazione delle Poste di Milano n. 163464 del 13/2/1963. Stampa: Staroffset s.r.l., Cernusco S/N (MI). Distribuzione SO.DI.P. S.p.A., Cinisello Balsamo (MI).

"ELETTRONICA DIGITALE"
si compone di
70 fascicoli settimanali
da suddividere
in 2 raccoglitori.

RICHIESTA DI NUMERI ARRETRATI. Per ulteriori informazioni, telefonare dal lunedì al venerdì ore 9.30-12.30 all'ufficio arretrati tel. 02/242021. Se vi mancano dei fascicoli o dei raccoglitori per completare l'opera, e non li trovate presso il vostro edicolante, potrete riceverli a domicilio rivolgendovi direttamente alla casa editrice. Basterà compilare e spedire un bollettino di conto corrente postale a PERUZZO & C. s.r.l., Ufficio Arretrati, viale Marelli 165, 20099 Sesto San Giovanni (MI). Il nostro numero di c/c postale è 42980201. L'importo da versare sarà pari al prezzo dei fascicoli o dei raccoglitori richiesti, più le spese di spedizione € 3,10 per pacco. Qualora il numero dei fascicoli o dei raccoglitori sia tale da superare il prezzo globale di € 25,82 e non superiore a € 51,65, l'invio avverrà per pacco assicurato e le spese di spedizione ammonteranno a € 6,20. La spesa sarà di € 9,81 da € 51,65 a € 103,29; di € 12,39 da € 103,29 a € 154,94; di € 14,98 da € 154,94 a € 206,58; di € 16,53 da € 206,58 in su. Attenzione: ai fascicoli arretrati, trascorse dodici settimane dalla loro distribuzione in edicola, viene applicato un sovrapprezzo di € 0,52, che andrà pertanto aggiunto all'importo da pagare. Non vengono effettuate spedizioni contrassegno. Gli arretrati di fascicoli e raccoglitori saranno disponibili per un anno dal completamento dell'opera. IMPORTANTE: è assolutamente necessario specificare sul bollettino di c/c postale, nello spazio riservato alla causale del versamento, il titolo dell'opera nonché il numero dei fascicoli e dei raccoglitori che volete ricevere.

IN REGALO in questo fascicolo

Modulo "Bread Board" per prove

IN REGALO nel prossimo fascicolo

- 1 Cavetto verde rigido
- 2 Transistor BC548 o BC547
- 2 Resistenze 1,8 K 5% 1/4 W
- 2 Resistenze 47 K 5% 1/4 W
- 2 Resistenze 1 M 5% 1/4 W
- 2 Resistenze 330 K 5% 1/4 W
- 2 Condensatori ceramici 22 nF

COME RACCOGLIERE E SUDDIVIDERE L'OPERA NELLE 4 SEZIONI

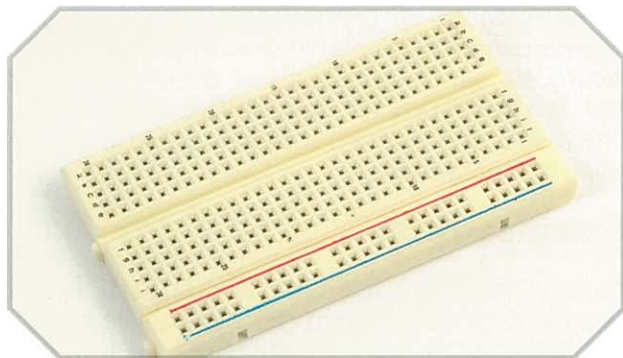
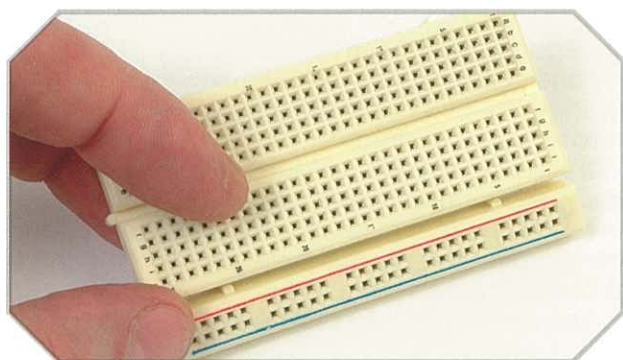
L'Opera è composta da 4 sezioni identificabili dalle fasce colorate, come indicato sotto. Le schede di ciascun fascicolo andranno suddivise nelle sezioni indicate e raccolte nell'apposito raccoglitore, che troverai presto in edicola. Per il momento, ti consigliamo di suddividere le sezioni in altrettante cartellette, in attesa di poterle collocare nel raccoglitore. A prima vista, alcuni numeri di pagina ti potranno sembrare ripetuti o sbagliati. Non è così: ciascuno fa parte di sezioni differenti e rispecchia l'ordine secondo cui raccogliere le schede. **Per eventuali domande di tipo tecnico scrivere al seguente indirizzo e-mail:** elettronicadigitale@microrobots.it

Hardware Montaggio e prove del laboratorio

Digitale di base Esercizi con i circuiti digitali

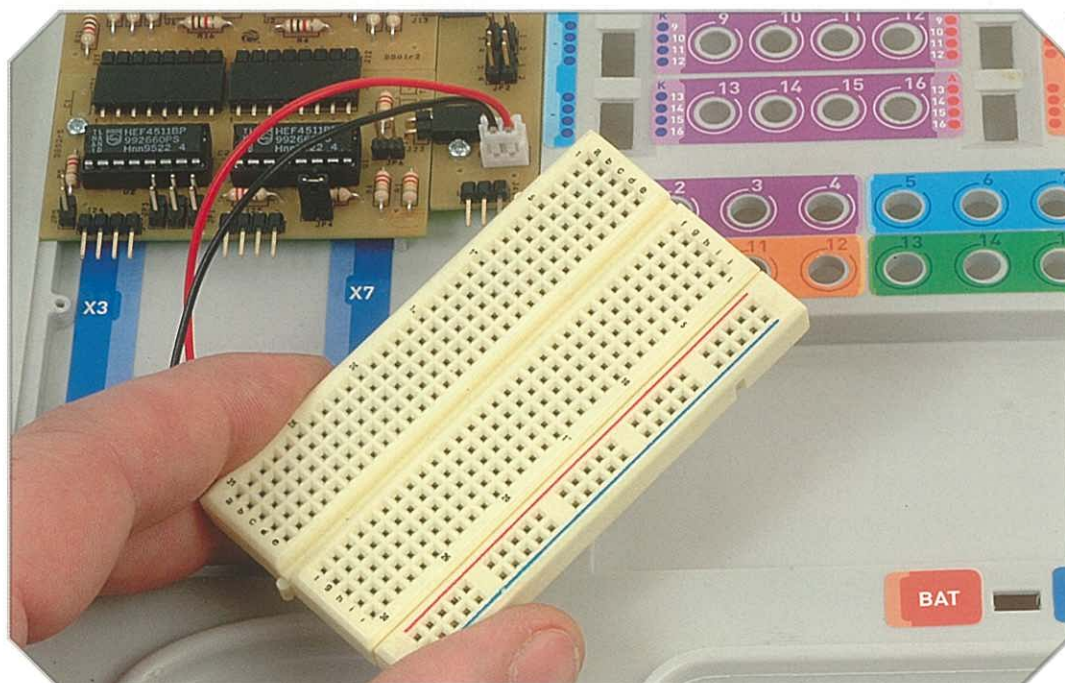
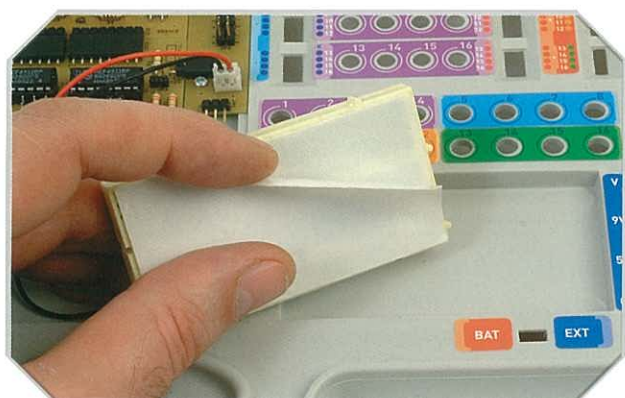
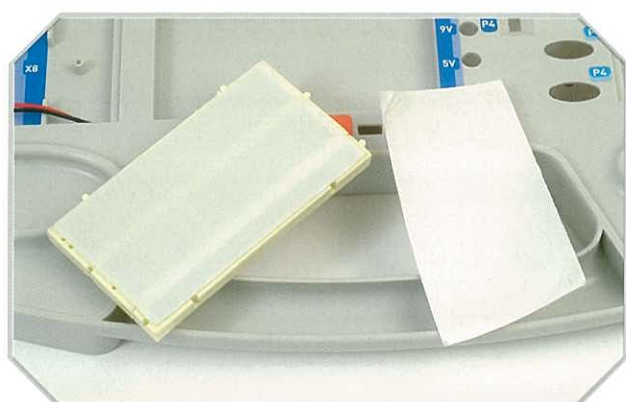
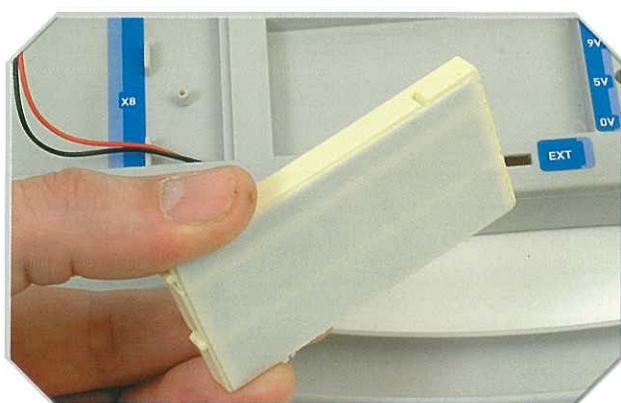
Digitale avanzato Esercizi con i circuiti sequenziali

Microcontroller Esercizi con i microcontroller

Modulo Bread Board

Modulo Bread Board.

Questo modulo è formato da due parti unite.

Il modulo Bread Board è uno dei componenti principali di questo laboratorio. Si tratta di un dispositivo di utilizzo universale, ovvero, può essere utilizzato per realizzare gli esercizi proposti con i componenti che vi forniremo oltre a molti altri esperimenti con questi o altri componenti, quindi questo laboratorio è un progetto aperto, pensato anche per i lettori professionisti dell'elettronica, attuali o futuri. Questo tipo di scheda si utilizza abitualmente nei corsi di formazione di elettronica per fare pratica, dato che permette di realizzare rapidamente e recuperare i componenti utilizzati.

Il modulo

Il modulo fornito è di qualità elevata, questo permette il suo utilizzo ripetuto senza che diminuiscano le sue caratteristiche qualitative di connettività. La qualità dei contatti metallici usati

Il laboratorio dispone di un apposito alloggiamento.

Il modulo ha un nastro adesivo con una lamina di protezione.

Vista posteriore, pronto per essere inserito.

Si prende ai bordi per non toccare l'adesivo.

come elementi di connessione, garantisce una buona conduttività elettrica e un collegamento sicuro, sia per la composizione del materiale sia per le caratteristiche meccaniche dello stesso, che assicurano anche il collegamento dal punto di vista meccanico.

Composizione

Questo modulo è formato da due parti, la più grande delle quali ha un canale centrale e trenta file da cinque fori per ogni lato. Questi cinque fori hanno, al loro interno, cinque punti di connessione uniti tra di loro. Se guardiamo la posizione in cui la scheda è montata sul laboratorio, vedremo che i terminali identificati come a, b, c, d, ed e sono collegati tra loro e sono indipendenti da quelli identificati come f, g, h, i e j, i quali sono anch'essi uniti tra loro. Riassumendo, questo modulo ha 300 punti di collegamento uniti tra loro a gruppi di cinque.

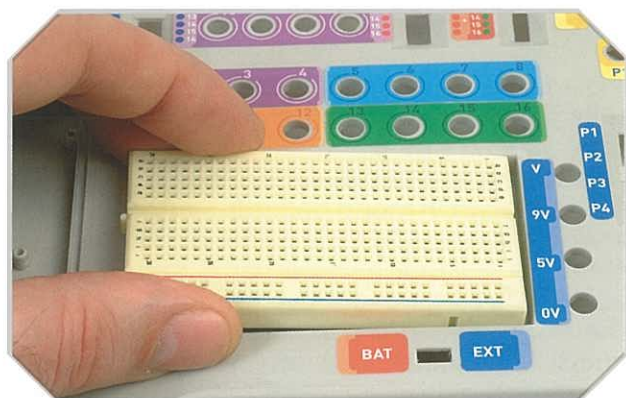
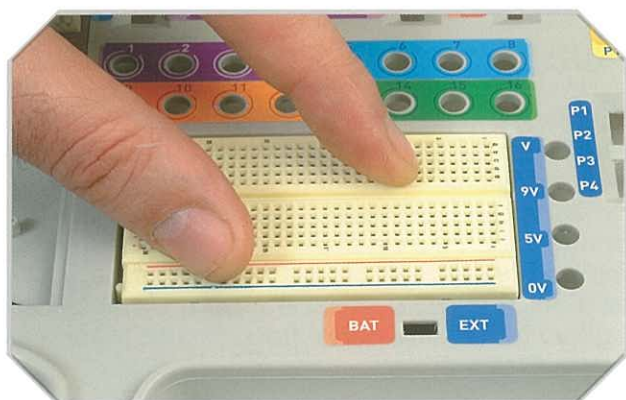
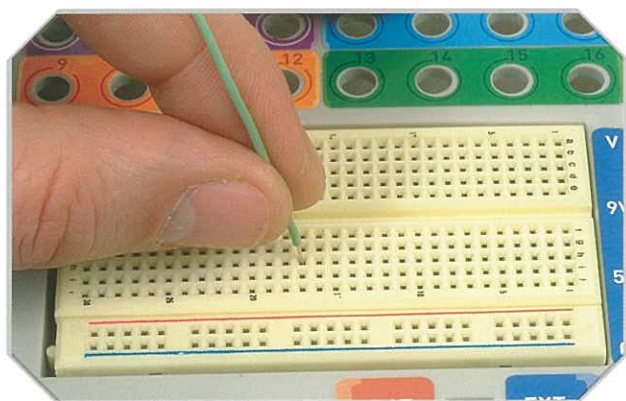
Il secondo modulo ha due file di collegamenti da 25 terminali ciascuna, uniti a gruppi di 25, come indicano le due linee colorate. Questi terminali si utilizzano normalmente per i collegamenti dell'alimentazione.

Entrambi i moduli vengono forniti già assemblati e uniti tra di loro.

Posizione

Sul pannello del laboratorio vi è una sede di circa sette millimetri di profondità, quindi la parte superiore del modulo, una volta inserito, ha un'altezza che facilita il lavoro col modulo stesso. La collocazione deve essere fatta in modo tale che il modulo di questa scheda, che contiene le due file da 25 terminali per i collegamenti dell'alimentazione, rimanga situato nella zona più vicina alla maniglia di trasporto del laboratorio.

Montaggio

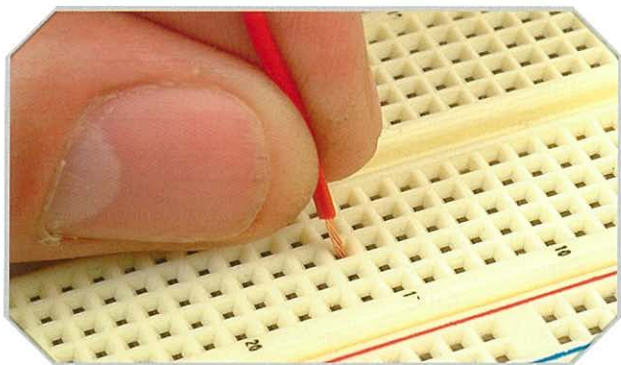
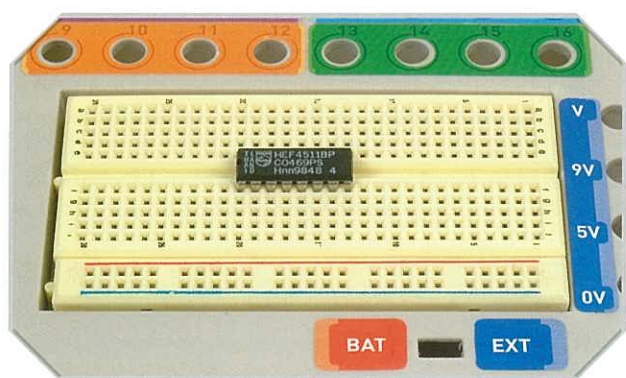
Il montaggio di questo modulo è piuttosto semplice, se guardiamo la sua parte posteriore vedremo che contiene una striscia di nastro biadesivo con uno dei lati già incollato al modulo stesso in modo da fissare le file di lamine con la funzione di connettore. L'altro lato del nastro biadesivo è protetto da una lamina asportabile. Questa lamina di protezione non deve essere tolta fino al momento in cui dovrà essere incollata, non dovrà quindi prendere aria per molto tempo né essere appoggiata su qualsiasi superficie, anche se lucida.

La scheda deve essere centrata prima che l'adesivo entri in contatto con il laboratorio.

Una leggera pressione è sufficiente per montare il modulo in modo definitivo.

Gli estremi dei cavi si inseriscono direttamente sul modulo.

Prima di eseguire qualsiasi operazione bisognerà assicurarsi che il pannello del laboratorio sia pulito, asciutto e che non abbia polvere. Inoltre dovremo anche controllare la posizione in cui verrà montato il modulo. Dopo aver realizzato queste verifiche, toglieremo la lamina di protezione dell'adesivo avendo cura di lasciare la parte adesiva di circa mezzo millimetro di spessore incollata al modulo, la lamina che viene tolta è invece molto sottile. Bisogna evitare di toccare con le dita la superficie adesiva per non sporcarla.

L'alloggiamento per il modulo, previsto sul pannello principale del laboratorio, è sufficientemente ampio per permettere una agevole installazione. Prima di incollarlo bisogna centrare bene il modulo nella sede, tenendo conto che l'adesivo è molto forte e non sarà possibile correggere un errore di posizione.

Se cercheremo di togliere il modulo una volta incollato, rovineremo l'adesivo e usciranno fuori le lamine di connessione; a questo punto, l'unica soluzione possibile sarà ricollocare ogni lamina al suo posto senza deformare i terminali e togliere tutto il resto dell'adesivo per poterne utilizzare dell'altro nuovo. Questa procedura, che è stata realizzata durante i lavori di progetto, è molto laboriosa, quindi vi consigliamo di realizzare il lavoro di montaggio con molta attenzione per non commettere errori.

Da ricordare

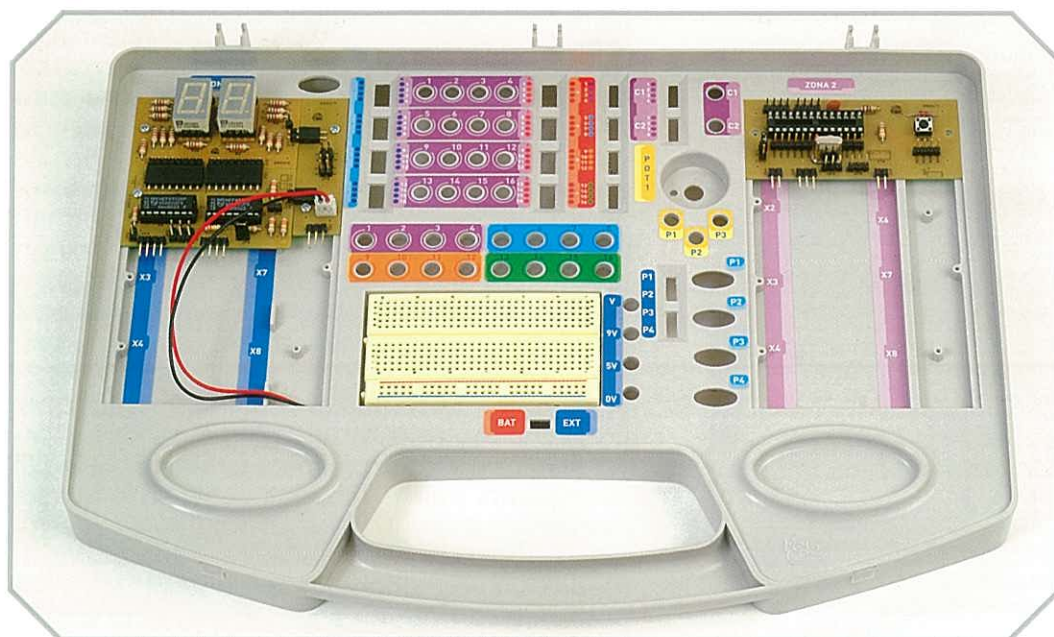
La qualità di questo modulo è determinata dall'elevata qualità dei suoi contatti. Per fare in modo che questi contatti mantengano le loro caratteristiche di connessione, occorre averne cura proteggendoli da due importanti nemici: la sporcizia e i terminali troppo grandi. La sporcizia può arrivare da polvere ambientale, che in alcune zone è presente in quantità; questo normalmente si evita chiudendo la scheda quando non la si utilizza, cosa che risulterà molto facile quando il laboratorio sarà completo, dato che si potrà chiudere come una valigetta da viaggio. Un'altra sorgente di sporco, che a volte passa inosservata, deriva dai terminali dei componenti, i quali devono essere controllati bene prima dell'inserzione. Eviteremo di limare o tagliare vicino al laboratorio, poiché i trucioli generati in queste operazioni possono finire sul modulo; se questo, però, risultasse inevitabile, capovolgeremo il modulo, invece di pulirlo con uno straccio, che potrebbe contribuire a far penetrare lo sporco nei fori del modulo stesso.

Si possono utilizzare anche cavi multifilo.

Gli integrati si inseriscono al centro del modulo.

Cavi e terminali

Il diametro raccomandato per il cablaggio dei collegamenti è di 0,5 mm, e in nessun caso si devono superare gli 0,8 mm; inoltre, per favorire l'inserzione, i terminali dei componenti devono essere tagliati in modo obliquo. L'estremo che si inserisce sul modulo deve essere diritto, pulito e libero da isolante almeno per 6 mm. È anche possibile utilizzare del cavo multifilo, sempre che il suo diametro si mantenga all'interno di questi limiti; inoltre sarà necessario allineare bene i reofori o ritorcerli leggermente fra loro.

Questo modulo è progettato per poter montare al centro i circuiti integrati in formato DIL. Quando si estrae l'estremo di un terminale di un cavo bisogna fare attenzione a non danneggiare il conduttore di rame tagliandolo, per evitare che si rompa e il terminale rimanga dentro al foro della scheda.

Pulizia

La polvere depositata si può pulire con un aspirapolvere. Se non ne disponiamo possiamo capovolgere il laboratorio in modo che il modulo rimanga con i fori verso il basso e pulirlo con un pennello da 1 o 2 cm di diametro. Non dovranno mai essere utilizzati prodotti per la pulizia, a base di silicone che si utilizzano normalmente per pulire i mobili, ma non si possono utilizzare per pulire il laboratorio.

Aspetto del laboratorio con i componenti forniti fino a questo momento.

Collegamenti del modulo Bread Board

Il modulo Bread Board facilita la realizzazione dei prototipi, prima di iniziare a lavorare, però, è necessario fare una serie di raccomandazioni per fare in modo che il lavoro realizzato sia il più proficuo possibile. Su questo modulo i terminali dei componenti si inseriscono a pressione, senza la necessità di realizzare delle saldature.

Lo schema

Prima di realizzare un circuito disegneremo il suo schema elettrico, in quanto non è conveniente affidarci unicamente alla memoria. Dopo aver realizzato lo schema lo studieremo con attenzione, per ottenere una buona distribuzione dei componenti sulla scheda e per verificare se parte del circuito non sia già disponibile sotto forma di modulo in un'altra zona del laboratorio.

I componenti

I componenti si montano e si collegano allo stesso tempo, è sufficiente appoggiare i terminali sul foro e spingere in modo che entrino a pressione tra le due lamine di contatto, mediante le quali si ottiene un buon fissaggio meccanico e un buon contatto elettrico.

In questo tipo di moduli i primi componenti che bisogna inserire sono i circuiti integrati, il formato DIL è il più utilizzato e il centro della scheda è riservato a essi.

I componenti a due o più terminali si montano in modo da sfruttare al massimo le possibilità della scheda; se, ad esempio, alcuni di questi terminali devono essere collegati a dei pin del circuito integrato, si utilizzerà uno dei quattro contatti che rimangono disponibili per ogni suo terminale.

I collegamenti

I collegamenti tra i componenti che vengono inseriti all'interno del modulo, si devono realizzare sfruttando al massimo le possibilità del modulo stesso. Quindi, per il positivo e il negativo dell'alimentazione si utilizzeranno le due file di terminali che dispongono, ognuna, di 25 contatti tutti collegati fra loro. Per gli altri contatti è necessario ricordare che per ogni fila i 5 terminali sono collegati fra loro. I terminali che non potranno essere uniti tra loro mediante i contatti del modulo, verranno uniti con un cavetto rigido da 0,5 mm di diametro.

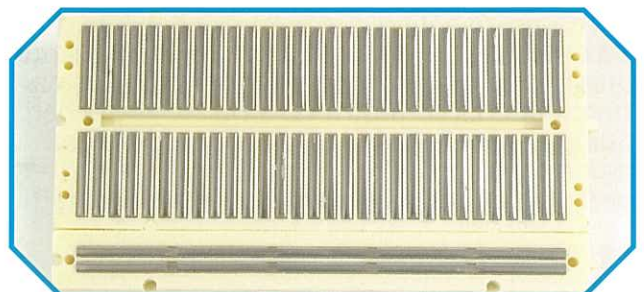
I contatti sono progettati per essere utilizzati con cavi o terminali di componente, di diametro compreso tra 0,3 e 0,8 mm.

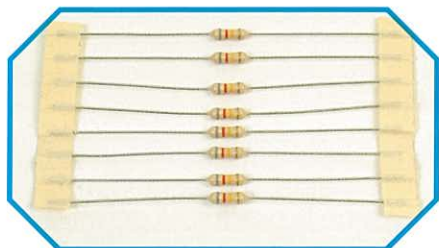
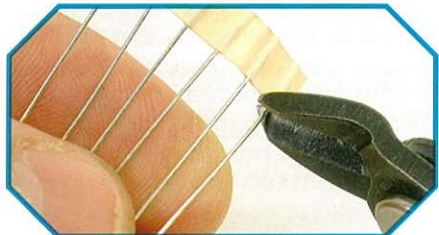
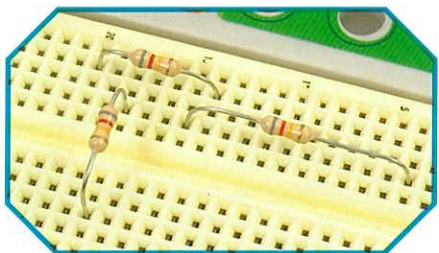
Il cablaggio

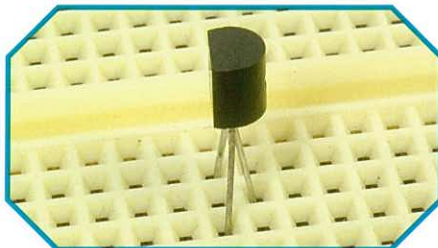
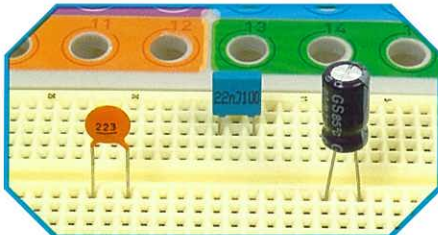
Per i collegamenti interni del modulo, e da questi verso l'esterno, si utilizzerà preferibilmente un filo rigido da 0,5 mm di diametro. Prima di utilizzarlo, bisognerà asportare circa 6 mm della copertura isolante da entrambe le estremità.

Gli integrati

Come abbiamo già detto, i circuiti integrati si inseriscono al centro della scheda, e bisogna tener presente che ogni pin dell'integrato occupa uno

Schema dei collegamenti interni del modulo Bread Board.

Per realizzare questa fotografia è stato tolto l'adesivo dal modulo. Osservate i collegamenti.

Dettaglio di una delle lamine di collegamento dei cinque terminali.

Il costruttore fornisce le resistenze in strisce, incollate a del nastro adesivo di carta.

I terminali devono rimanere ben diritti, il taglio obliquo facilita l'inserzione

Esempio di resistenze inserite.

Transistor correttamente inserito, con ogni terminale collegato a una fila indipendente.

Transistor collegato male, i suoi terminali sono collegati insieme essendo inseriti sulla stessa fila.

Campioni di condensatori.

Gli integrati si inseriscono al centro della scheda.

dei 5 collegamenti disponibili di ogni fila, lasciando, quindi, quattro possibilità per collegare direttamente cavi o terminali di componenti a ogni pin dell'integrato.

Le resistenze

Le resistenze hanno due terminali e la loro pulizia deve essere curata con attenzione, in quanto questi componenti sono forniti incollati a due strisce laterali di nastro di carta, ci potrebbero quindi essere residui di colla che dovranno essere asportati o, semplicemente, si potrà tagliare il pezzo superiore per evitare di sporcare i contatti.

I transistor

Per utilizzare i transistor bisogna allargare leggermente i loro terminali per adattarli alla distanza standard fra i fori del modulo, che è un decimo di pollice, ovvero 2,54 mm. Devono essere inseriti in modo tale che ogni terminale occupi una fila differente, avendo cura di identificare ogni terminale, ovvero collettore, base ed emettitore.

Montaggio

Il montaggio deve seguire questo ordine: per prima cosa si installano i circuiti integrati, poi i transistor, i condensatori, i diodi e le resistenze. Bisogna sfruttare al massimo i collegamenti interni del modulo.

Dopo aver montato tutti i componenti si iniziano a realizzare le connessioni, che non è stato possibile fare direttamente, mediante dei pezzi di filo rigido da 0,5 mm di diametro spelati ad entrambe le estremità.

Compilazione con MPLAB

Ci sono errori nel nostro programma? Funzionerà come vogliamo? Cosa succede se cambio questa istruzione con un'altra? Tutte queste domande possono trovare una risposta su MPLAB.
Finora abbiamo editato un programma, abbiamo visto la configurazione e una piccola introduzione, dobbiamo però addentrarci in questo software e iniziare a gestire alcune delle sue funzioni più importanti.

Prima di compilare

Per lavorare utilizzeremo l'esercizio del capitolo precedente di MPLAB. Dobbiamo aprire il nostro progetto selezionando sul menù di controllo Project → Open Project.

Sceglieremo l'ultimo progetto che abbiamo fatto all'interno della cartella di lavoro e cliccheremo OK. Ci apparirà un messaggio di errore il quale ci indica che non esiste alcun file in codice macchina associato al progetto.

Accetteremo l'errore e apparirà la videata con l'editor e il codice che abbiamo preparato la volta precedente. Il passo successivo sarà quello di tornare a editare il nostro progetto selezionando sul menù di controllo Project → Edit Project, con la conseguente comparsa della finestra che possiamo vedere nella figura della pagina successiva. Se clicchiamo sul file che appare con l'estensione tra parentesi quadre, in questo caso è ese2[.hex], si attiverà la possibilità di selezionare Node Properties (prima era disabilitato).

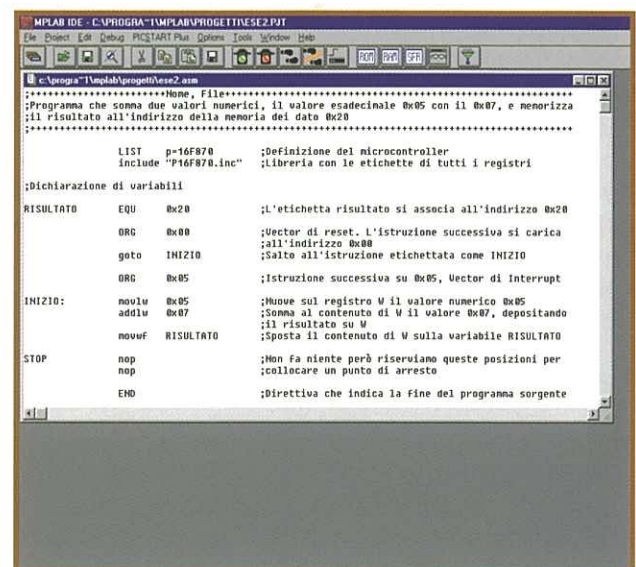
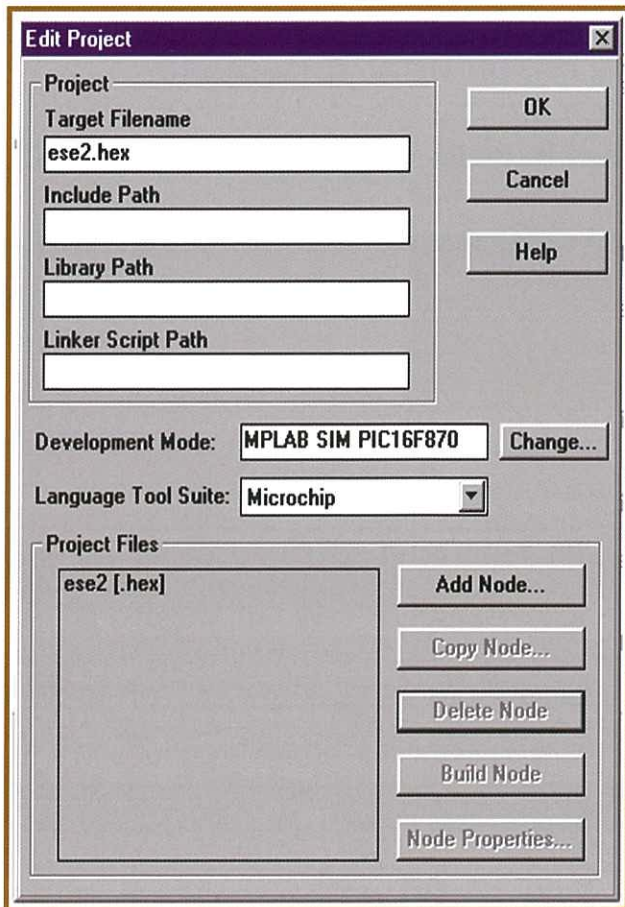
Scegliendo questa opzione accederemo a una finestra da dove si selezioneranno i file e i formati che si otterranno assemblando il programma. Non occorre cambiare nulla, dobbiamo solo sapere che abbiamo selezionato la creazione di un file di errore (error file) durante la procedura di assemblamento, un file con il listato del programma (list file) e abbiamo impostato la sensibilità all'utilizzo delle maiuscole (case sensitivity). Cliccheremo OK per continuare e torneremo alla videata precedente in cui selezioneremo Add Node.

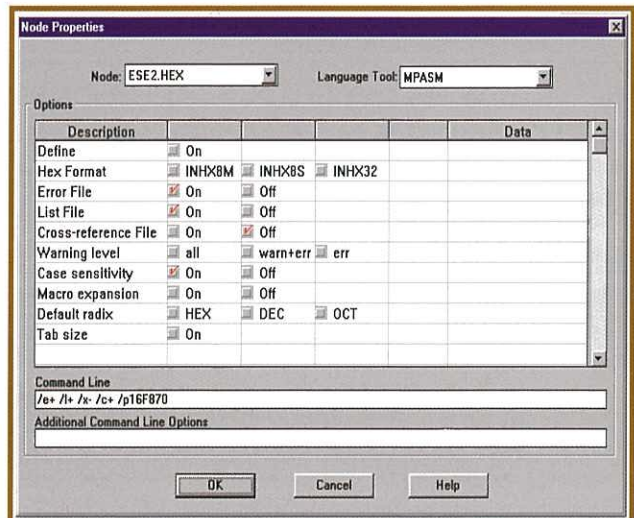
Nella finestra che appare sceglieremo il file .asm contenente il codice necessario per sommare i due numeri che erano stati proposti come esercizio nel capitolo precedente. In seguito potremo verificare che nel campo Project Files è stato aggiunto il file selezionato. Cliccheremo OK e torneremo alla videata con il programma scritto. Siamo pronti per la compilazione.

Import Error

No hex file has been built for this project

OK

Messaggio del programma all'apertura del progetto.

Partiamo dall'esercizio di somma di due numeri.

Videata di edizione del progetto.

Messaggio di errore causato dalla mancanza del file assembler all'interno del progetto.

Proprietà del nodo per l'assemblaggio.

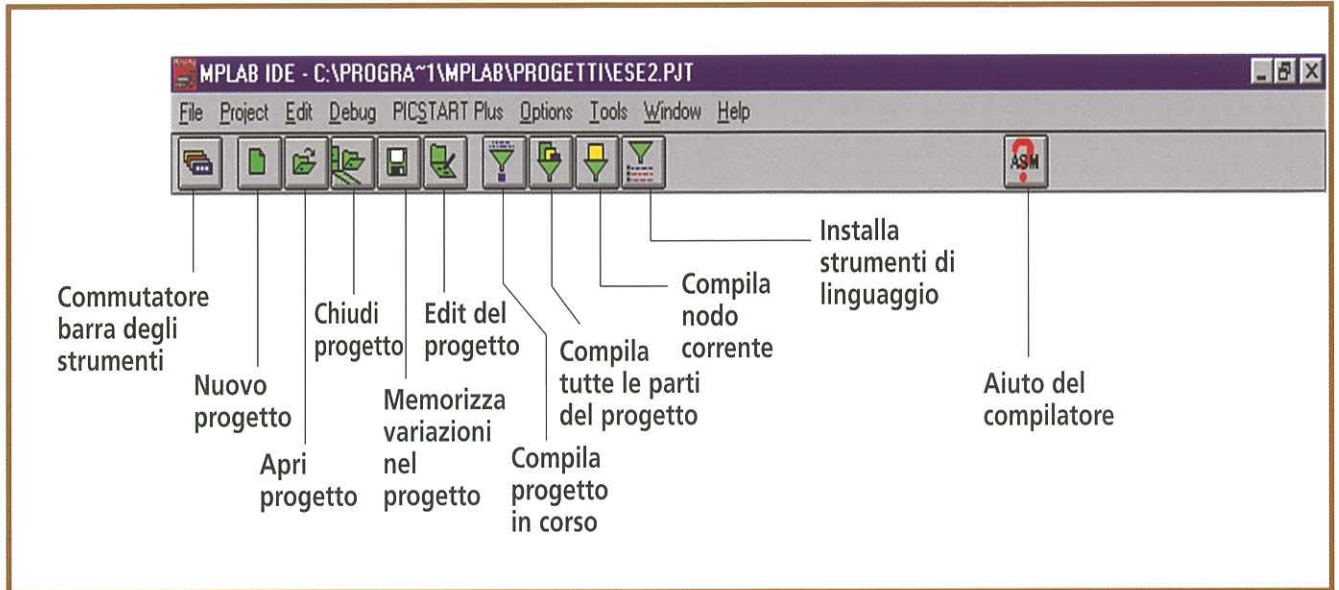
Strumenti di compilazione

Possiamo commutare tra diverse barre di strumenti in MPLAB, una delle quali è specifica per il lavoro con i progetti e la compilazione. Nella figura della pagina successiva possiamo vedere la barra degli strumenti di compilazione a cui facevamo riferimento con tutte le sue funzioni in dettaglio. Vi consigliamo di realizzare nuovamente tutti i passaggi visti nel capitolo precedente, utilizzando però i pulsanti della barra degli strumenti.

La compilazione

Se non abbiamo eseguito i passaggi precedenti, in cui abbiamo aggiunto il file in assembler al progetto cercando di assemblare, apparirà il messaggio di errore della figura. MPLAB ci indica mediante questo messaggio che non è stato specificato alcun file come sorgente.

Noi abbiamo seguito tutti i passaggi, quindi potremo compilare direttamente. Per assemblare il programma dobbiamo selezionare nel menù di controllo l'opzione Project → Build All o, utilizzando la barra degli strumenti, cliccare il pulsante corrispondente a Compilazione congiunta del progetto.

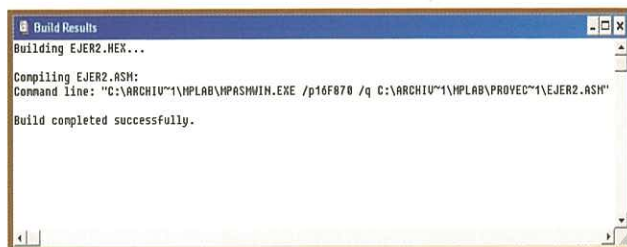
Quando un programma non ha errori, il risultato della compilazione sarà quello indicato nella figura. Apparirà una videata in cui ci verrà comunicato che la compilazione è stata eseguita con successo. A partire da questo momento, nella directory di lavoro dove si trovavano il progetto e il file in assembler, si troveranno anche il file con il listato del codice, il file degli errori e il file con il codice macchina (estensione .hex). Quest'ultimo file sarà quello da scrivere sul PIC e farà girare su di esso l'applicazione che abbiamo creato. Utilizzeremo sempre Build All, dato che questa opzione include la compilazione di tutta la struttura

Barra degli strumenti di progetti e compilazione.

del progetto. Il resto delle opzioni Make Project e Build Node non contemplano l'intero progetto.

Gli errori

Quando programmiamo, una delle domande che ci poniamo è: cosa succede quando in un programma ci sono degli errori?

Se in un programma ci sono errori MPLAB li rileva e impedisce la compilazione. Inseriremo alcuni errori nel nostro programma. Nella figura possiamo osservare come siano stati introdotti tre errori: l'omissione del punto e virgola su una linea che volevamo fosse un commento, l'inserimento di un simbolo nell'operando che indica un indirizzo e la presenza di un carattere nello mnemonico di un'istruzione.

Quando si realizza una modifica all'interno del codice, si devono salvare i cambiamenti prima di eseguire la compilazione. Quindi

Videata del risultato di una compilazione senza errori.

selezioneremo Project → Save Project o il pulsante della barra degli strumenti corrispondente, preparandoci per compilare. Quando lanceremo la compilazione, il risultato di questa azione è una videata come quella mostrata nella figura della pagina successiva. Questa videata indica la presenza di errori e comunica che la compilazione non ha avuto luogo.

Se ci posizioniamo con il mouse sulla prima linea degli errori e clicchiamo due volte, il cur-

```
c:\progra~1\mplab\progetti\ese2.asm
;++++++++++++++++++++++Nome, File++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;Programma che somma due valori numerici, il valore esadecimale 0x05 con il 0x07, e memorizza
;il risultato all'indirizzo della memoria dei dato 0x20
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            LIST    p=16F870          ;Definizione del microcontroller
            include "P16F870.inc"     ;Libreria con le etichette di tutti i registri

Dichiarazione di variabili                                    ERRORI

RISULTATO   EQU     0x20              ;L'etichetta risultato si associa all'indirizzo 0x20

            ORG     0x00              ;Vector di reset. L'istruzione successiva si carica
                                      ;all'indirizzo 0x00
            goto    INIZIO            ;Salto all'istruzione etichettata come INIZIO

            ORG     0x05$             ;Istruzione successiva su 0x05, Vector di Interrupt

INIZIO:     movlaw  0x05              ;Muove sul registro W il valore numerico 0x05
            addlw   0x07              ;Somma al contenuto di W il valore 0x07, depositando
                                      ;il risultato su W
            movwf   RISULTATO         ;Sposta il contenuto di W sulla variabile RISULTATO

STOP        nop                       ;Non fa niente però riserviamo queste posizioni per
            nop                       ;collocare un punto di arresto

            END                       ;Direttiva che indica la fine del programma sorgente
```

Errori inseriti nel codice.

```
Build Results
Building ESE2.HEX...

Compiling ESE2.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F870 /q C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM"
Error[122]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 9 : Illegal opcode (di)
Error[113]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 15 : Symbol not previously defined (INIZIO)
Error[112]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 17 : Missing operator
Error[122]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 19 : Illegal opcode (movlaw)
Error[118]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 20 : Overwriting previous address contents (0000)
Error[118]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 20 : Overwriting previous address contents (0000)

MPLAB is unable to find output file "ESE2.HEX". This may be due to a compile, assemble, or link process failure

Build failed.
```

Risultato finale del tentativo di compilazione.

```
Build Results
Building ESE2.HEX...

Compiling ESE2.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F870 /q C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM"
Error[113]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 15 : Symbol not previously defined (INIZIO)
Error[112]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 17 : Missing operator
Error[122]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 19 : Illegal opcode (movlaw)
Error[118]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 20 : Overwriting previous address contents (0000)
Error[118]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 20 : Overwriting previous address contents (0000)

MPLAB is unable to find output file "ESE2.HEX". This may be due to a compile, assemble, or link process failur

Build failed.
```

Dopo aver corretto il primo errore rimangono da correggere gli altri.

sore si posizionerà sulla linea di codice che contiene l'errore. Il messaggio che appare nella finestra spiega in che cosa consiste l'errore, anche se è in inglese, specifica cos'è che non va. In questo primo errore informa che è stato trovato un simbolo non definito in precedenza, una parola sconosciuta. Posizionandoci con il cursore su questa linea e con l'aiuto del messaggio che ci fornisce il compilatore, possiamo identificare chiaramente l'errore e correggerlo. Torneremo quindi a scrivere il punto e virgola all'inizio, salveremo il programma e proveremo nuovamente a compilare. Questa volta vedremo che gli errori si sono ridotti, ora il compilatore ci informa della presenza di cinque errori.

Se ripetiamo la stessa operazione con il primo errore inserito, vedremo che prima dell'etichetta di INIZIO c'è un simbolo strano. Questo non permette il riconoscimento dell'etichetta. Per risolvere togliamo questo simbolo, salviamo nuovamente e torniamo a compilare. Gli errori si sono ridotti a due. A volte il compilatore, in presenza di un errore, ci informa di tutto ciò che trova di estraneo o non valido per la compilazione. Per questo, un semplice errore come quello corretto in precedenza può provocare tre rilevamenti di errore da parte del compilatore o tre ragioni che impediscono al programma di essere compilato correttamente.

Osserviamo i due errori rimanenti, vedremo chiaramente che abbiamo utilizzato uno mnemonico sbagliato. Correggendo l'errore torneremo alla situazione iniziale in cui il programma si compila correttamente e, quindi, genera i file necessari per caricare il programma sul microcontroller.

Conclusioni

Applicare questi nuovi concetti servirà come esercizio. Sarà possibile creare un progetto con il codice dell'esercizio precedente e cercare di compilare senza inserire il file assembler all'interno del progetto. Se inserirete degli errori nel programma potrete familiarizzare con il modo di cui dispone MPLAB di indicare gli errori; in definitiva, è consigliabile provare tutto ciò che serve per affinare le nuove conoscenze acquisite.

```
Build Results
Building ESE2.HEX...

Compiling ESE2.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F870 /q C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM"
Error[113]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 15 : Symbol not previously defined (INIZIO)
Error[122]   C:\PROGRA~1\MPLAB\PROGETTI\ESE2.ASM 19 : Illegal opcode (movlaw)

MPLAB is unable to find output file "ESE2.HEX". This may be due to a compile, assemble, or link process failur

Build failed.
```

Dopo aver corretto il secondo errore il compilatore rileva solamente due errori.

Composizione di un programma

Indipendentemente dal linguaggio di programmazione che si utilizza, alcune caratteristiche sono comuni in tutti i programmi. Possono esistere alcune differenze tra le parti che compongono un programma, l'ordine di queste, l'obbligatorietà dell'utilizzo o la loro denominazione, però possiamo raggrupparle per la loro funzione e, in questo modo, renderle comuni per qualsiasi linguaggio.

Routine

Un programma deve avere come minimo una routine di esecuzione e, benché dipenda dalla complessità dell'applicazione, normalmente sarà composta anche da diverse subroutine o sottoprogrammi. In una subroutine o sottoprogramma si inserisce un codice che risolve un'azione specifica. Ogni subroutine contiene istruzioni che sono organizzate in strutture di controllo. Sia le subroutine che le istruzioni e le strutture di controllo richiedono, per il loro funzionamento, l'utilizzo di variabili e costanti.

Altri elementi comuni a qualsiasi programma sono gli organigrammi che, come già sappiamo, sono rappresentazioni visive dello sviluppo dei programmi stessi.

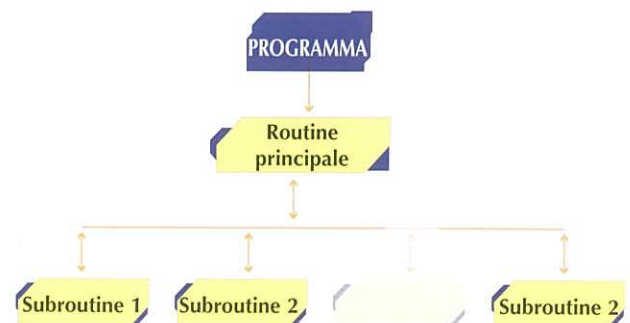
Variabili e costanti

Una variabile è un contenitore che ospita un valore, il quale può essere cambiato lungo il corso del programma. Identificheremo la variabile, che potrà avere valori diversi nel tempo, con un nome.

Una costante, invece, conterrà un valore che sarà sempre uguale. Daremo un nome anche alla costante, però il valore contenuto in essa non verrà mai modificato.

L'utilizzo di variabili e di costanti facilita molto la programmazione, evitando di far riferimento a valori specifici e possibili confusioni nello sviluppo del codice. Ad esempio, immaginate un programma che risolva uno sviluppo matematico in cui utilizzeremo il valore pi (3,14159...). Se definiamo una costante e la chiamiamo PI e a essa assegniamo il valore 3,14159..., ogni volta che dovremo utilizzarlo nel programma faremo riferimento a PI e non al suo valore.

Immaginate ora un controllo di temperatura in cui un sensore registra i valori di questa ogni dieci secondi. Ogni minuto presenteremo sul display la media delle temperature. Immaginate la propensione agli errori che risulterebbe in un codice in cui si operi con tutti i valori acquisiti in forma numerica. Risulta molto più semplice definire una variabile dove scrivere questi valori e successivamente operare sulla variabile.

Sia le variabili che le costanti possono essere di diverso tipo, secondo la classe di valore che contengono. Non è la stessa cosa contenere un numero decimale invece che un testo, quindi in molti linguaggi di programmazione è necessario specificare di quale tipo sono le variabili e le costanti. A seconda del tipo si utilizzerà più o meno spazio per memorizzare il contenuto. In alcuni linguaggi di programma-

Composizione generale di un programma.

Campioni	1	2	3	4	5	6	Operazione
Valore	22,4	22.6	22.6	22.5	22.3	22.1	$\frac{22,4+22,6+22,6+22,5+22,3+22,1}{6}$
Variabile	Temp	Temp+1	Temp+2	Temp+3	Temp+4	Temp+5	$\frac{\sum Temp}{6}$

Esempio di utilizzo delle variabili per semplificare le operazioni.

Tipi di dati	Esempio di valori	Esempio di dichiarazione
Numeri esadecimali	4D	tempo EQU 4Dh
Numeri binari	10110001	valore EQU b'10110001'
Numeri decimali	26	int var=26
Simboli	HOLA	string messaggio='HOLA'
Dispositivi	16F870	LIST P=16F870

Tipi di dati ed esempi di dichiarazione.

zione è necessario inizializzare la variabile con un valore, anche se nel corso del programma verrà modificato.

Istruzioni

Le istruzioni sono il modo che abbiamo di indicare al processore di realizzare un'azione specifica. Dalla corretta definizione dell'istruzione dipenderà l'esecuzione dell'azione nel modo desiderato. All'interno di un programma un'istruzione ha un nome specifico (mnemonico), dei parametri e un determinato ordine e fa riferimento a un'azione.

Nei linguaggi di alto livello un'istruzione può indicare al processore di realizzare diversi lavori, invece, nei linguaggi di basso livello vicini al linguaggio macchina, le istruzioni non contengono operazioni multiple.

```
MPLAB IDE - C:\PROGRA~1\MPLAB\PROGETTI\PROVA2.PJT

        LIST    p=16F870                ;Definizione del microcontroller
        include "P16F870.inc"           ;Librerie con le etichette di tutti i registri

;Dichiarazione di variabili

RISULTATO       EQU     0x20            ;L'etichetta risultato si associa alla posizione 0x20

                ORG     0x00            ;Indirizzo di reset. L'istruzione succesiva si carica
                                        ;nell'indirizzo 0x00
                goto    INIZIO          ;Salto all'istruzione etichettata come INIZIO

                ORG     0x05            ;Istruzione successiva su 0x05, Vector di Interrupt

INIZIO          mxxovl  0x05            ;Sposta sul registro W il valore diretto 0x05
                addlw   0x07            ;Somma al contenuto di W il valore 0x07, depositando
                                        ;il risultato su W
                movwf   RISULTATO       ;Sposta il contenuto di W alla variabile RISULTATO

STOP            nopera                  ;Non fa nulla però riserviamo questo indirizzo per
                nop                     ;posizionare un punto di arresto

                END                     ;Direttiva che indica la fine del programma sorgente

Build Results
Building ESER2.HEX...

Compiling ESER2.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /p16F870 /q C:\PROGRA~1\MPLAB\PROGETTI\ESER2.ASM"
Error[113]   C:\PROGRA~1\MPLAB\PROGETTI\ESER2.ASM 11 : Symbol not previously defined (INIZIO)
Error[122]   C:\PROGRA~1\MPLAB\PROGETTI\ESER2.ASM 15 : Illegal opcode (mxxovl)
Error[122]   C:\PROGRA~1\MPLAB\PROGETTI\ESER2.ASM 20 : Illegal opcode (nopera)

MPLAB is unable to find output file "ESER2.HEX". This may be due to a compile, assemble, or link process f

Build failed.
```

Errori che rileva il compilatore a causa di un errato utilizzo delle istruzioni.

```
Forno75-125 - Blocco note

        LIST P=16F870
        include "P16F870.INC"

volt2           EQU     b'01111111'     ;a partire da 125° fissa l'uscita e inizia la sequenza
volt3           EQU     b'01001011'     ;a partire da 75° segnala la temperatura

aux1            EQU     0x30
aux2            EQU     0x31
aux3            EQU     0x32
aux4            EQU     0x33
aux5            EQU     0x34
aux6            EQU     0x35
n_pezzi         EQU     0x36
cont            EQU     0x37
cont1           EQU     0x38
aux11           EQU     0x39
aux21           EQU     0x3A
aux31           EQU     0x3B
aux41           EQU     0x3C
aux51           EQU     0x3D
```

Esempio di dichiarazione di costanti e variabili in assembler.

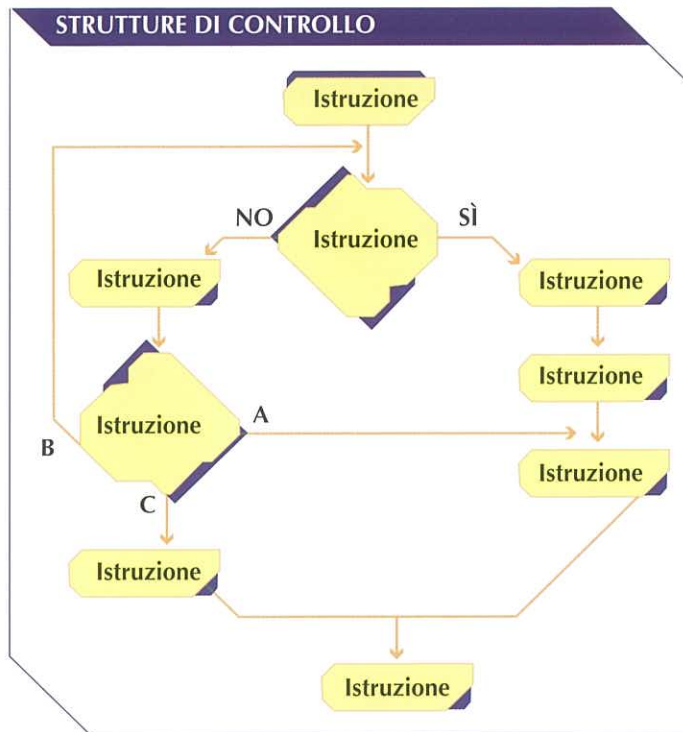
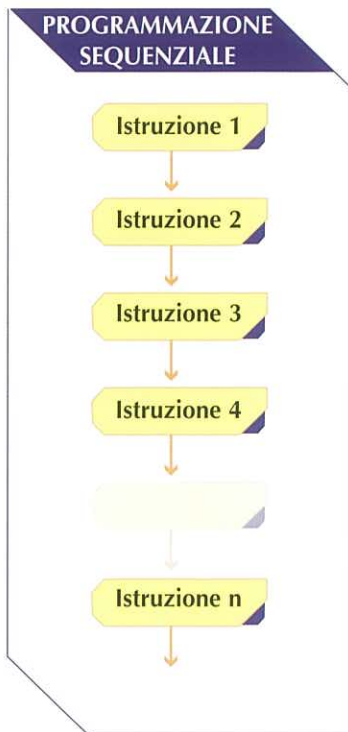
Abbiamo visto in precedenza che quando ci sbagliamo con il mnemonico dell'istruzione o con gli operandi di questa, si produce un errore che impedisce la compilazione del programma. Un errore si produrrà anche se, utilizzando le istruzioni, creeremo una struttura di controllo sbagliata, dobbiamo fare, invece, molta attenzione a quando non commettiamo un errore di questo tipo, ma utilizziamo le istruzioni in modo non corretto.

Immaginate il microcontroller che controlla un ascensore. Un utente all'interno dell'ascensore preme il pulsante di un piano, arriva al piano e poi si aprono le porte; non sarebbe la stessa cosa se l'utente preme il pulsante, si aprono le porte e poi si va al piano.

Bisogna fare particolare attenzione con gli errori semantici, anche se molti di essi possono essere corretti utilizzando gli organigrammi.

Strutture di controllo

Quando sviluppiamo un programma dobbiamo porre le istruzioni in un ordine preciso. Se non esistessero le strutture di controllo, un programma sarebbe formato da una serie di istruzioni messe una dietro l'altra, in modo sequenziale (verrebbero eseguite le istruzioni per ordine solamente una volta). Le strutture di controllo permettono al programmatore di realizzare comparazioni, creare strutture condizionali o alternative in modo che venga eseguita una serie di istruzioni solamente se si compie una determinata condizione o in presenza di un determinato evento. L'utilizzo di queste strutture ottimizza l'utilizzo dello spazio e del tempo nei programmi.

Diversi modi di sviluppare un programma.

Attualmente non si utilizza la programmazione sequenziale e tutti i linguaggi accettano la programmazione basata su strutture di controllo.

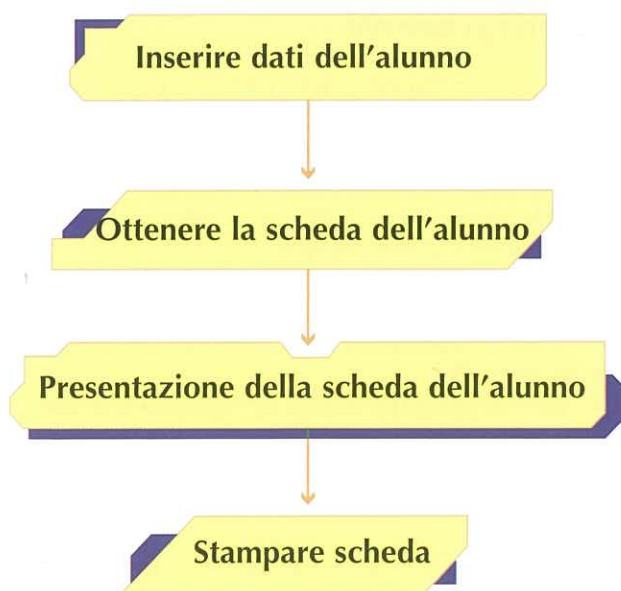
Normalmente un programma disporrà di diversi rami o percorsi, perché, anche se all'inizio non sappiamo come verrà eseguito, dobbiamo contemplare tutte le possibili opzioni.

Ogni linguaggio avrà differenti forme per esprimere le diverse strutture di controllo (condizionali, interattive, a risposta multipla...), però l'utilizzo di queste strutture risulta comune e fondamentale in tutti i linguaggi.

Inserire dati dell'alunno

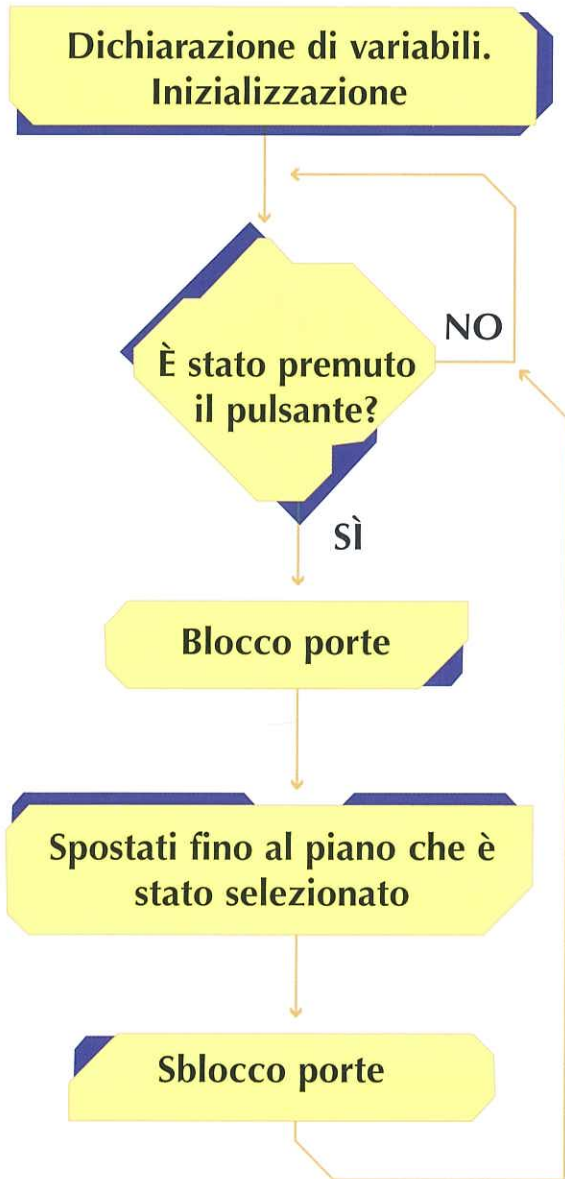
Ottenere la scheda dell'alunno

Presentazione della scheda dell'alunno

Stampare scheda

Un programma può essere composto da diverse subroutine.

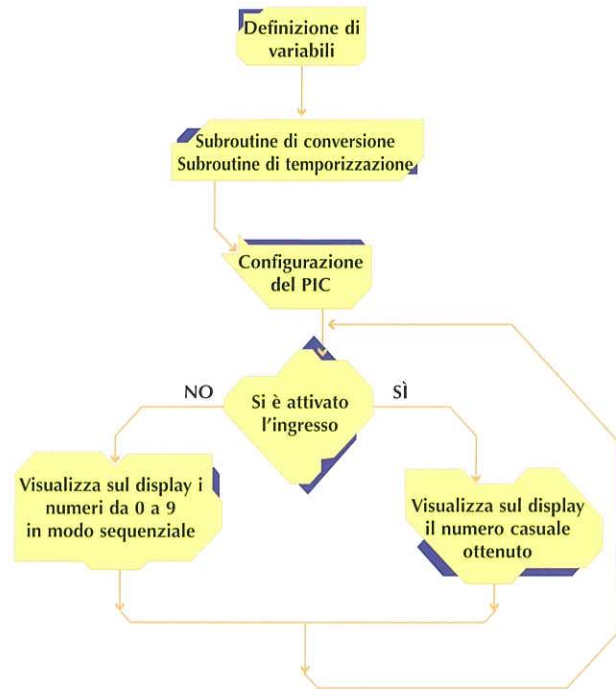
Subroutine

Sappiamo che le istruzioni si organizzano perché vengano eseguite mediante strutture di controllo, ma che succede quando il programma è molto lungo o ci sono parti che devono essere ripetute in diversi punti? In questo caso i frammenti di programma che sviluppano una funzione specifica vengono raggruppati sotto un nome e si separano dal programma principale ricorrendo a essi quando è necessario. Questi raggruppamenti di codice sono conosciuti come subroutine, procedimenti o funzioni, a seconda del linguaggio di programmazione che stiamo utilizzando.

Quando si desidera eseguire questa parte del programma non sarà più necessario scriverla per intero nuovamente, ma dovremo

L'utilizzo di organigrammi implica una miglior disposizione nel programma delle istruzioni ed evita di commettere errori semantici.

semplicemente far riferimento al suo nome, chiamando così in esecuzione la subroutine.

Molti programmatori hanno l'abitudine di lasciare la minor quantità possibile di codice nel programma principale, in modo da ridurlo a semplici chiamate a subroutine e lasciare che queste realizzino il lavoro.

Utilizzando le subroutine evitiamo di scrivere più codice del necessario, dato che non si producono ripetizioni di codice e si ottimizza la strutturazione del programma, rendendo più semplice la messa a punto e le successive modifiche.

Un altro organigramma.
Strumento fondamentale per un programmatore.

Organigrammi

Anche se non fanno parte del programma, gli organigrammi sono molto utili al programmatore. Un organigramma è la rappresentazione grafica di ciò che fa il programma e sarà la prima cosa che il programmatore crea prima di iniziare a sviluppare il codice.

Ripassiamo le regole per confezionare un organigramma:

— Ha un solo punto di ingresso e un solo punto di uscita.

— Un rettangolo indica un'azione che deve essere eseguita.

— Un rombo segnala una condizione e permette di seguire diversi percorsi.

— Mediante le frecce si ordinano le istruzioni.

— Le frecce che escono da un rombo devono essere etichettate per scegliere l'alternativa in modo corretto.